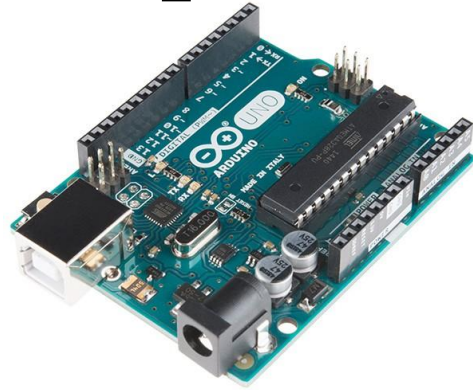


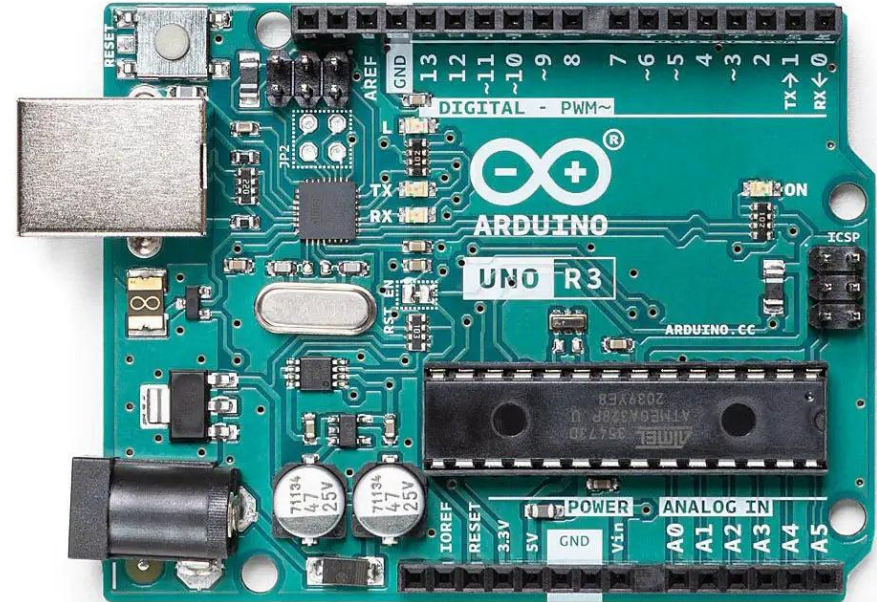
# Introducción a Arduino (express)



Antonio Rodríguez Nieto

# Qué es Arduino Uno?

- Plataforma de software libre
- Microcontrolador
- Reprogramable
- Basado en C++
- I/O (Entradas/salidas) configurables



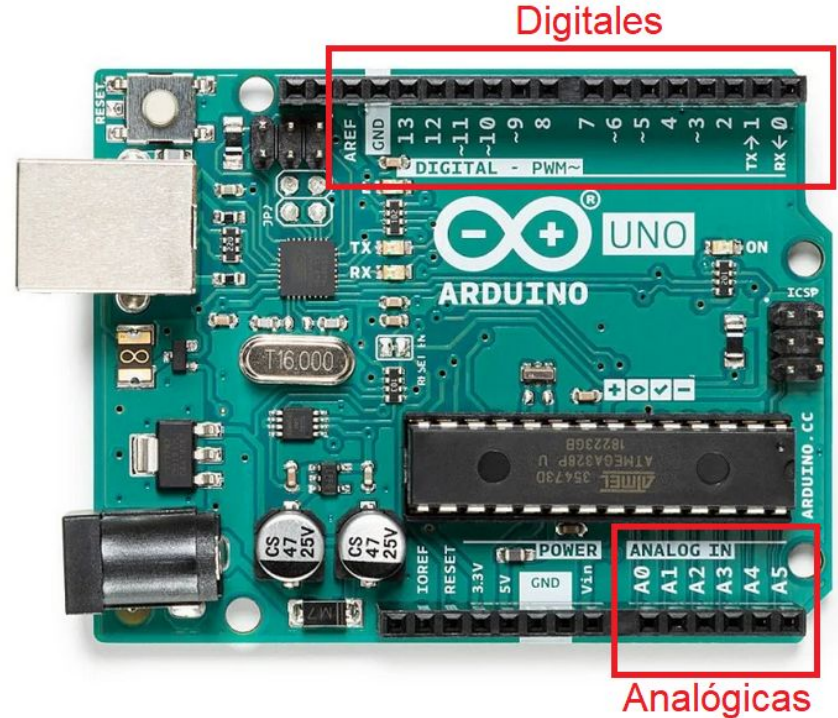
# Pines Arduino Uno

Se pueden encontrar dos tipos de pines:

- Digitales (entrada y salida):

Además, existen seis pines (~) que se pueden convertir en salidas analógicas “falsas”.

- Analógicos (entrada)

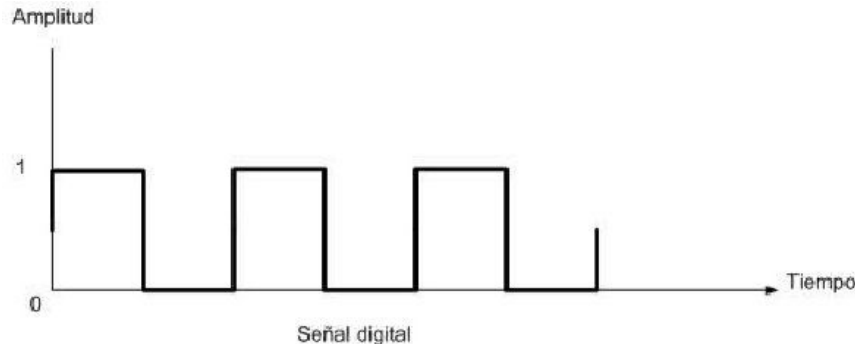


# Señales digitales

Las señales digitales se caracterizan por tener únicamente dos estados: BAJO o ALTO (LOW o HIGH).

Para actuadores, su uso suele ser en aquellos que únicamente tengan dos estados, encendido o apagado (una alarma, una bombilla...).

Para sensores, sería el mismo principio (detectar una presencia, un contacto...).



# Arduino IDE

El IDE (Entorno de Desarrollo Integrado) de Arduino cuenta con varias funcionalidades, siendo las más importantes:

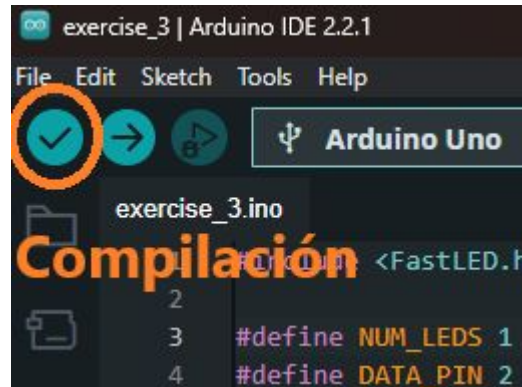
- Creación y compilación de código
- Portar (subir) a la placa de Arduino el programa compilado
- Visualización del puerto Serial en tiempo real



# Herramientas Arduino IDE

Las herramientas más importantes del IDE son las siguientes:

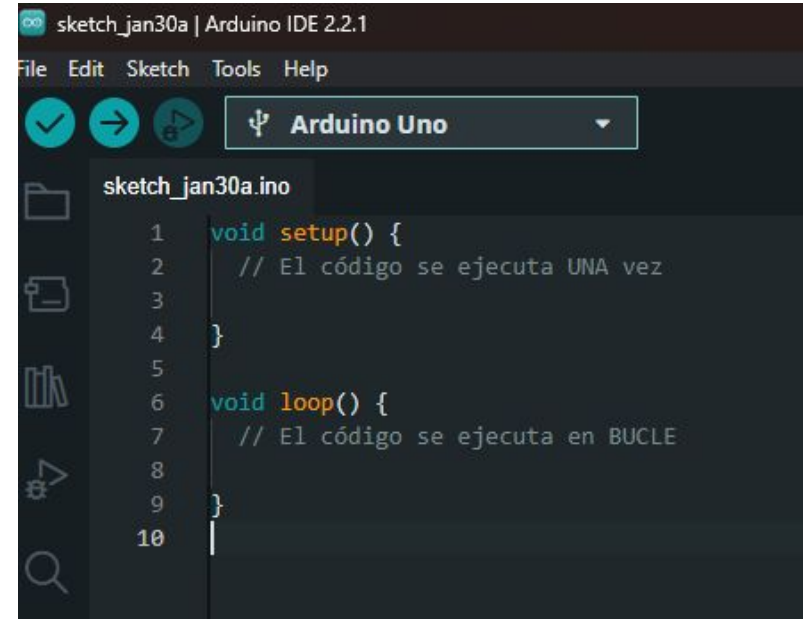
- Selección de placa (una vez conectada)
- Compilación
- Subir a la placa (si no se ha compilado, lo hace, y luego sube)



# Programación en Arduino IDE

La programación en Arduino se hace por defecto en las funciones `setup()` y `loop()`.

- `Setup()`: Se ejecuta únicamente una vez al inicio del programa. Utilizada para hacer configuración o inicializaciones.
- `Loop()`: Se ejecuta repetidamente una vez se ha terminado la función `setup()`. Es el núcleo del programa.



```
sketch_jan30a | Arduino IDE 2.2.1
File Edit Sketch Tools Help
[Check] [Run] [Upload] [Board: Arduino Uno]

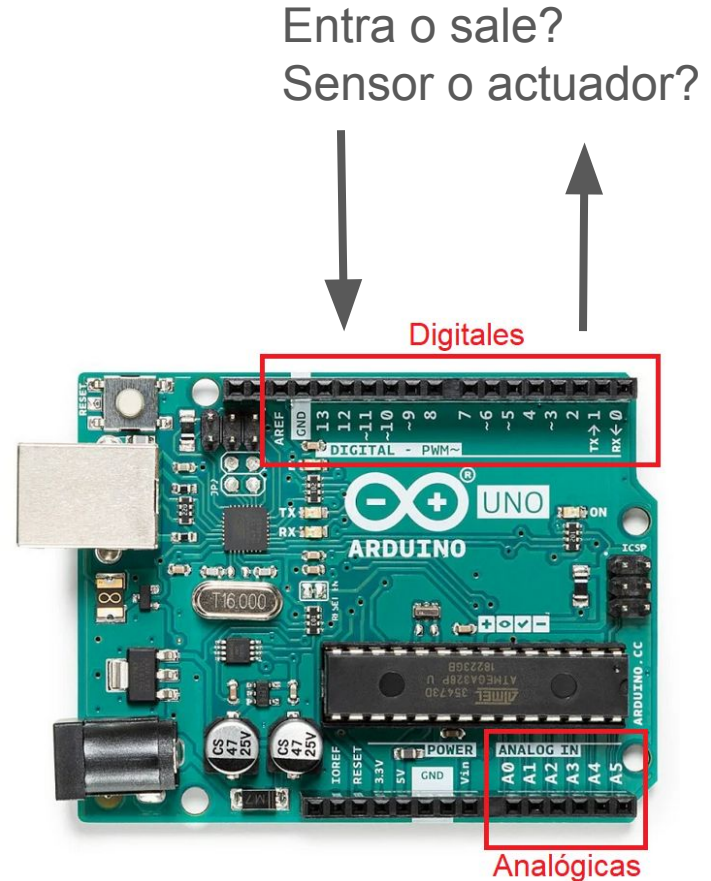
sketch_jan30a.ino
1 void setup() {
2     // El código se ejecuta UNA vez
3
4 }
5
6 void loop() {
7     // El código se ejecuta en BUCLE
8
9 }
10
```

# Programación en Arduino IDE

Para decirle a Arduino si un pin es de entrada o salida, se usa la función:

```
pinMode( pin, modo);
```

- `pinMode( 5, INPUT);`  
// pin 5 como entrada
- `pinMode( 10, OUTPUT);`  
// pin 10 como salida





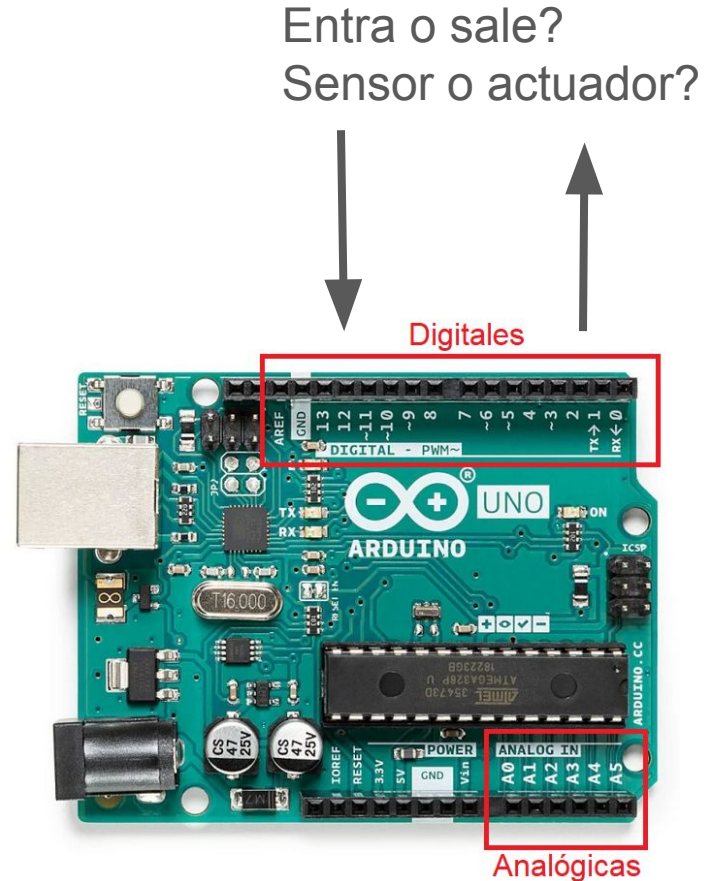
# Programación en Arduino IDE

Para leer o escribir a través de los pines, se usan las siguientes funciones:

- `digitalWrite(pin, estado):`

```
digitalWrite(10, HIGH);  
// 5V en pin 10  
digitalWrite(10, LOW);  
// 0V en pin 10
```

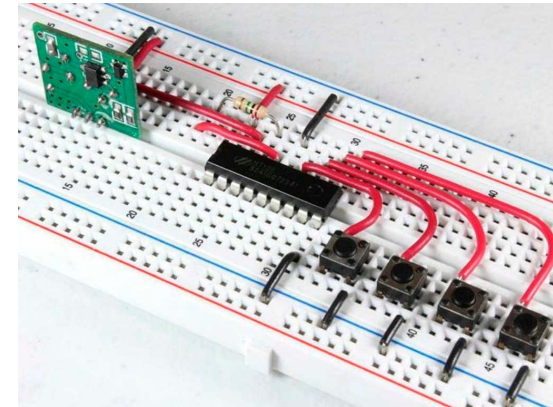
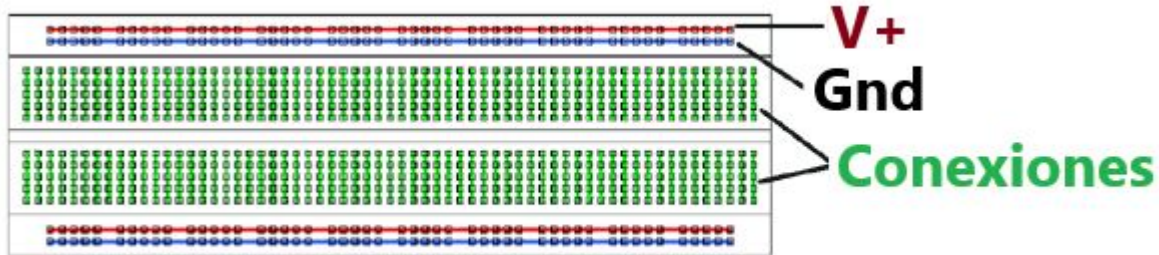
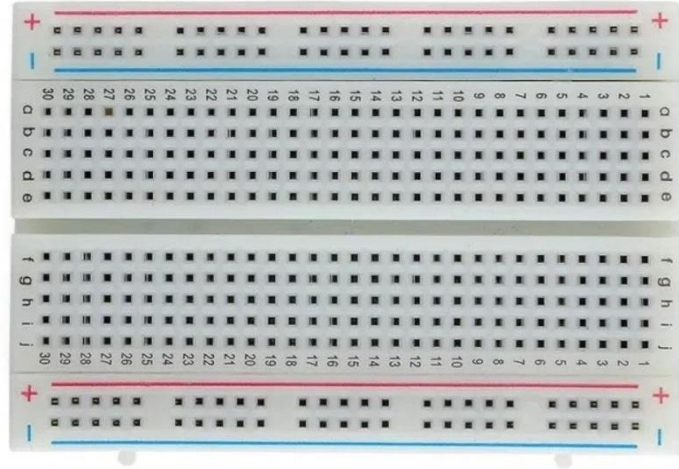
- `digitalRead(pin):`  
`int value = digitalRead(5);`  
// Lee el valor que tiene el pin 5  
(HIGH 5V o LOW 0V)



# Placa de prototipos

Utilizada para conectar dispositivos

Compuesta por vías interconectadas



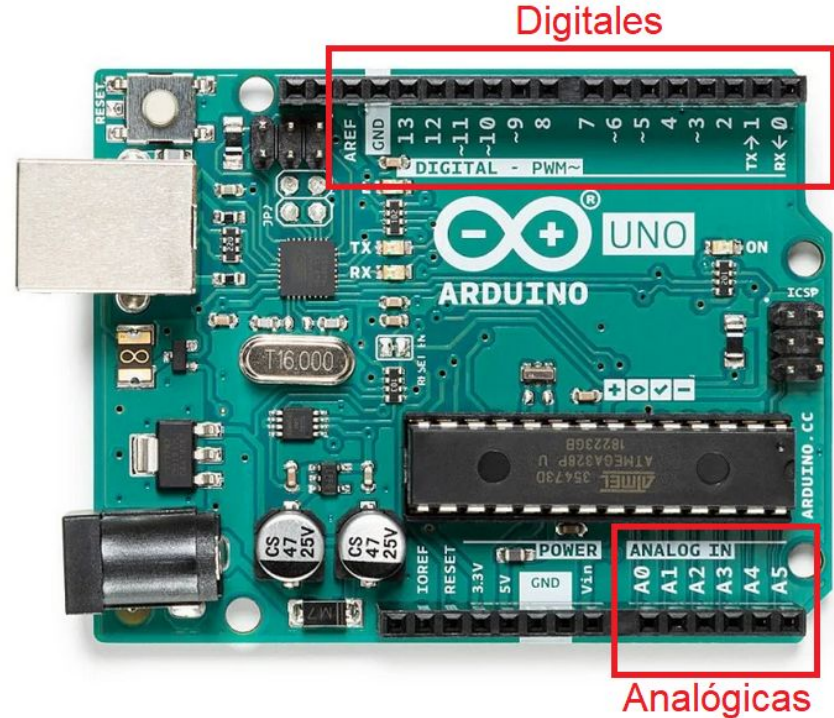
# Pines Arduino Uno

Se pueden encontrar dos tipos de pines:

- Digitales (entrada y salida):

Además, existen seis pines (~) que se pueden convertir en salidas analógicas “falsas”.

- Analógicos (entrada)

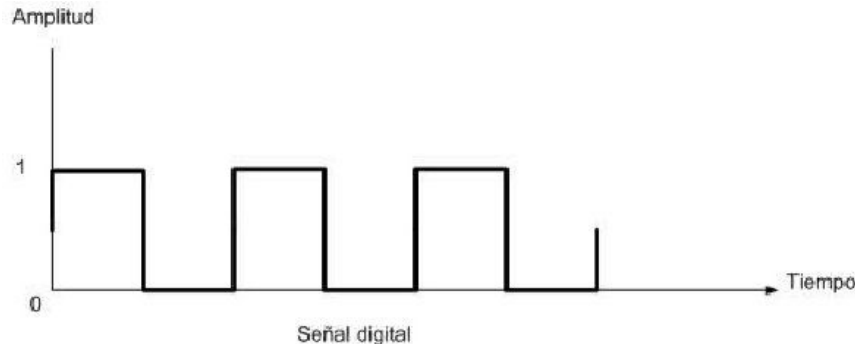


# Señales digitales

Las señales digitales se caracterizan por tener únicamente dos estados: BAJO o ALTO (LOW o HIGH).

Para actuadores, su uso suele ser en aquellos que únicamente tengan dos estados, encendido o apagado (una alarma, una bombilla...).

Para sensores, sería el mismo principio (detectar una presencia, un contacto...).

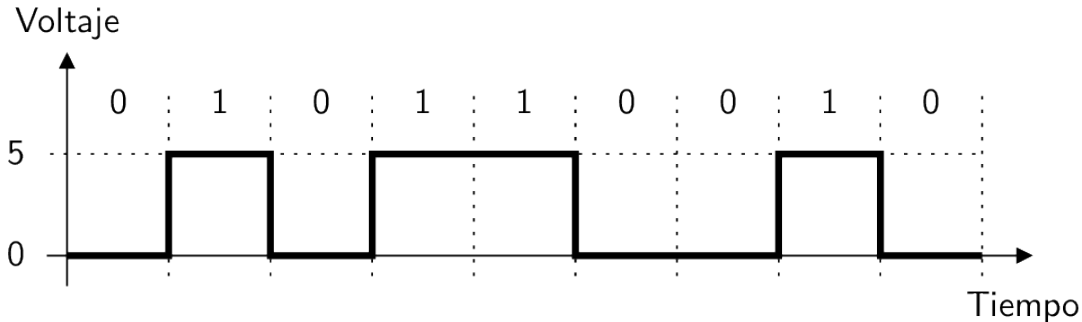


# Señales digitales

En el caso de Arduino Uno, un pin digital puede ser configurado como entrada (sensor) o salida (actuador).

Los niveles de voltaje digitales son:

- ALTO (HIGH) = 5V. En caso de “entrada”,  $> 3V$ .
- BAJO (LOW) = 0V. En caso de “entrada”,  $< 3V$ .



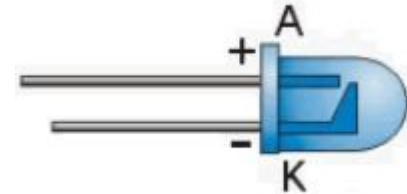
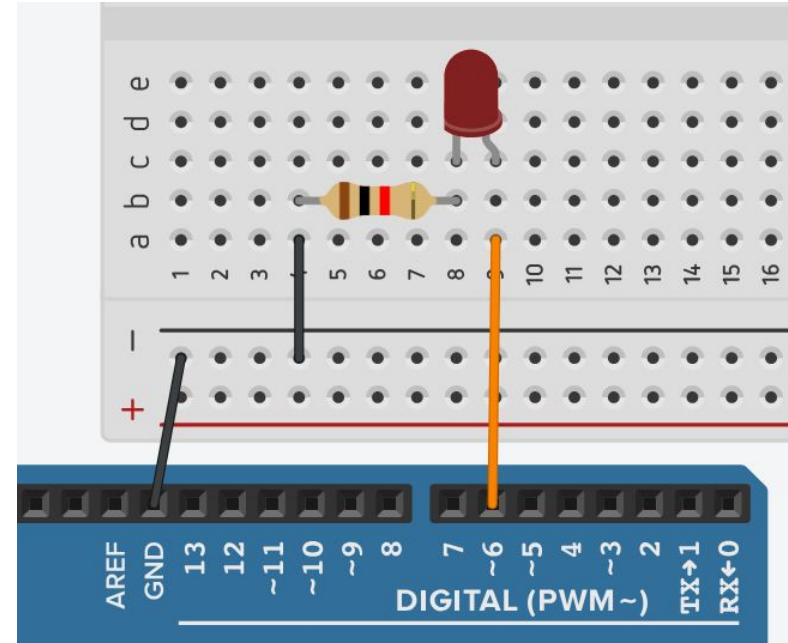
# Actividad

## Encender LED:

- Pata larga a pin digital
- Pata corta a resistencia
- Resistencia a GND

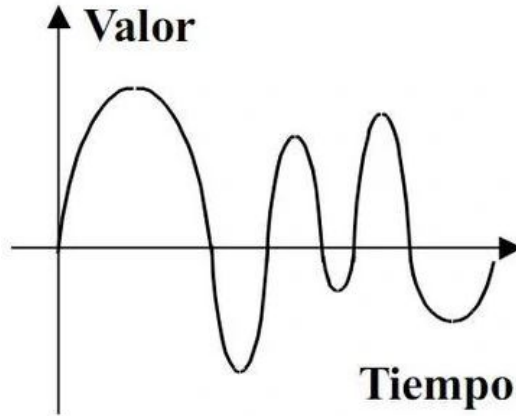
## Funciones:

- pinMode
- digitalWrite



# Señales analógicas

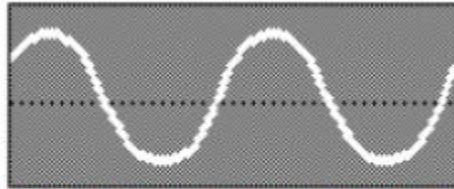
Estas señales, a diferencia de las digitales, tienen un rango de valores infinitos. Para su lectura, se convierte la señal analógica a digital.



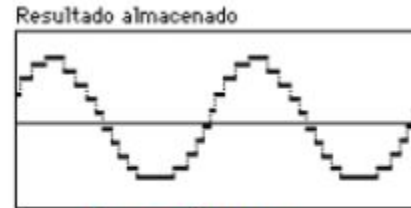
# Señales analógicas

En Arduino uno se pueden leer en los pines analógicos hasta 5V.

Como tenemos 10 bits de resolución, tenemos  $2^{10}=1024$  posibles niveles. Esto supone una precisión de  $\pm 2,44$  mV (5 V/1024 bits).



Señal analógica



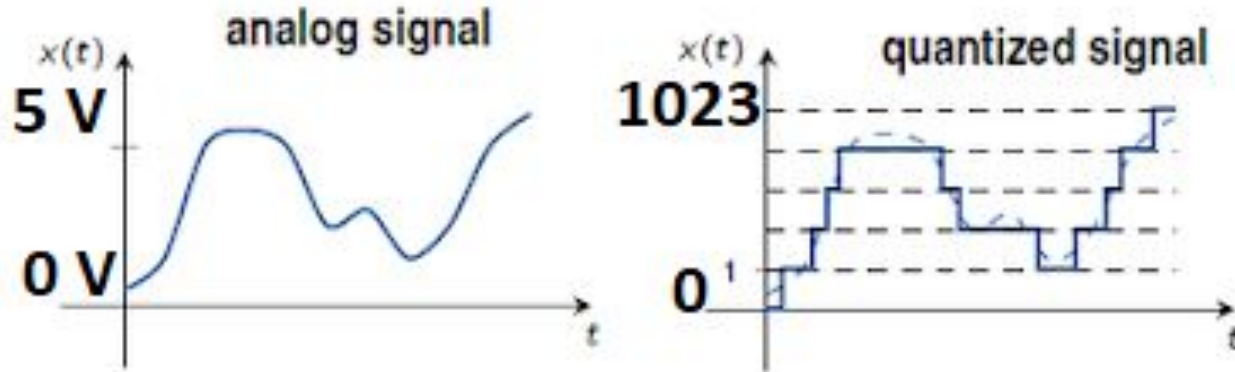
Lectura



# Señales analógicas

En Arduino uno se pueden leer en los pines analógicos hasta 5V.

Como tenemos 10 bits de resolución, tenemos  $2^{10}=1024$  posibles niveles. Esto supone una precisión de  $\pm 2,44 \text{ mV}$  (5 V/1024 bits).



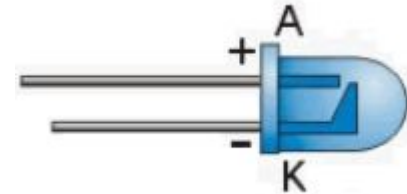
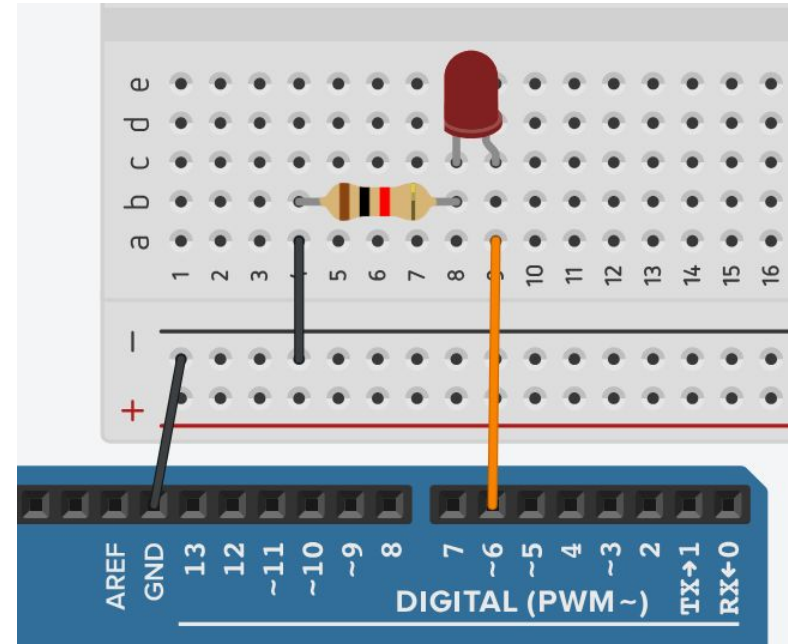
# Actividad

Variar luminosidad LED:

- Pata larga a pin digital
- Pata corta a resistencia
- Resistencia a GND

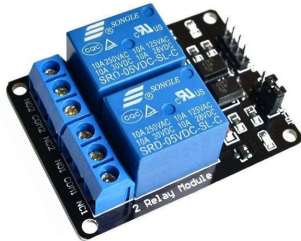
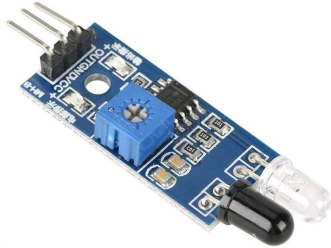
Funciones:

- pinMode
- analogWrite
- for

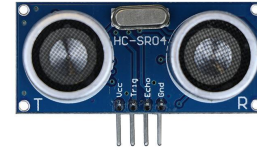


# Sensores y actuadores

## Digitales

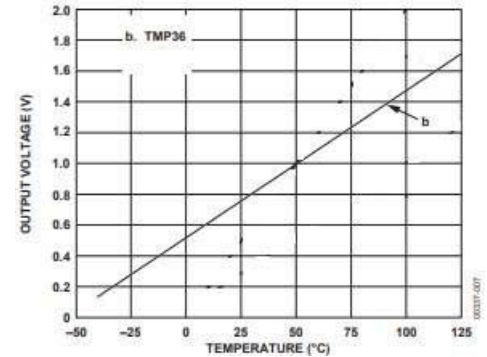
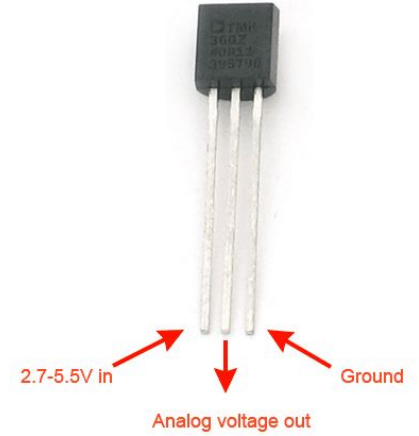
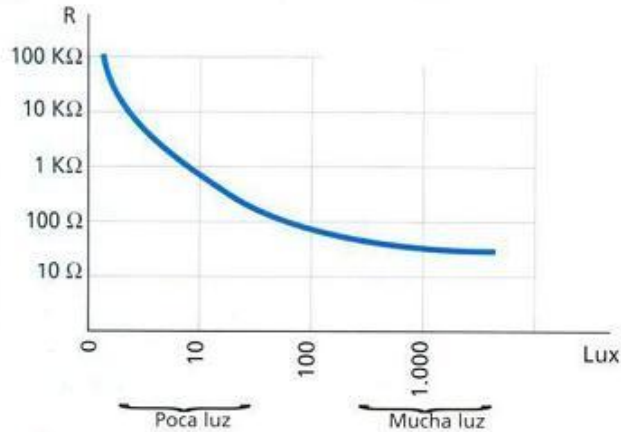
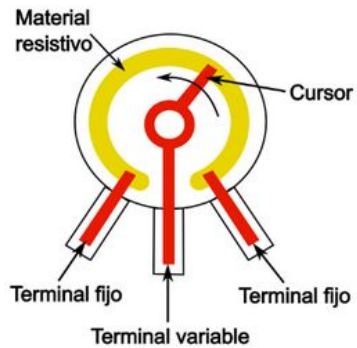


## Analógicos



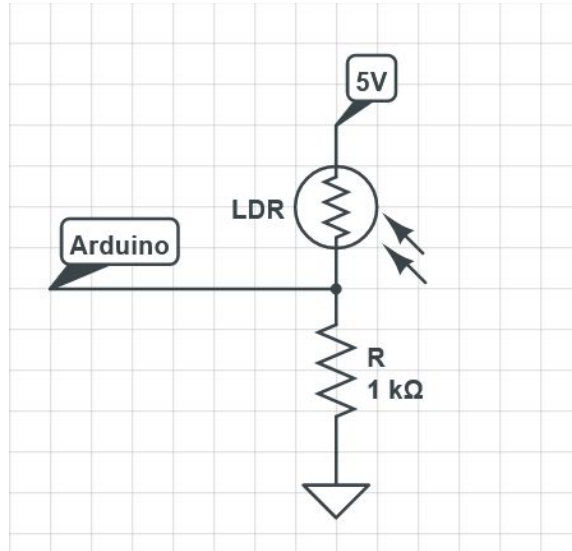
# Sensores

## Analógicos



# Lectura de sensores

La forma más sencilla de medir sensores de resistencia variable son mediante los divisores de voltaje:



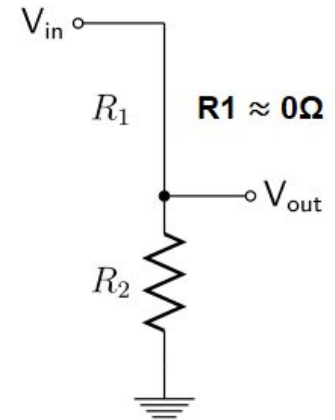
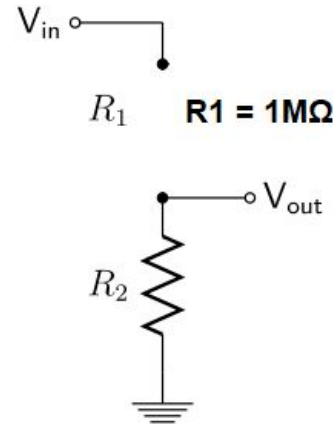
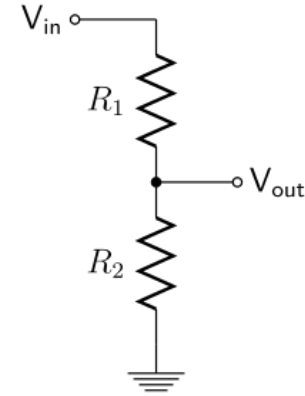
# Divisor de voltaje

Un divisor de voltaje funciona tal que :

$$V_{out} = V_{in} * \frac{R_2}{R_1 + R_2}$$

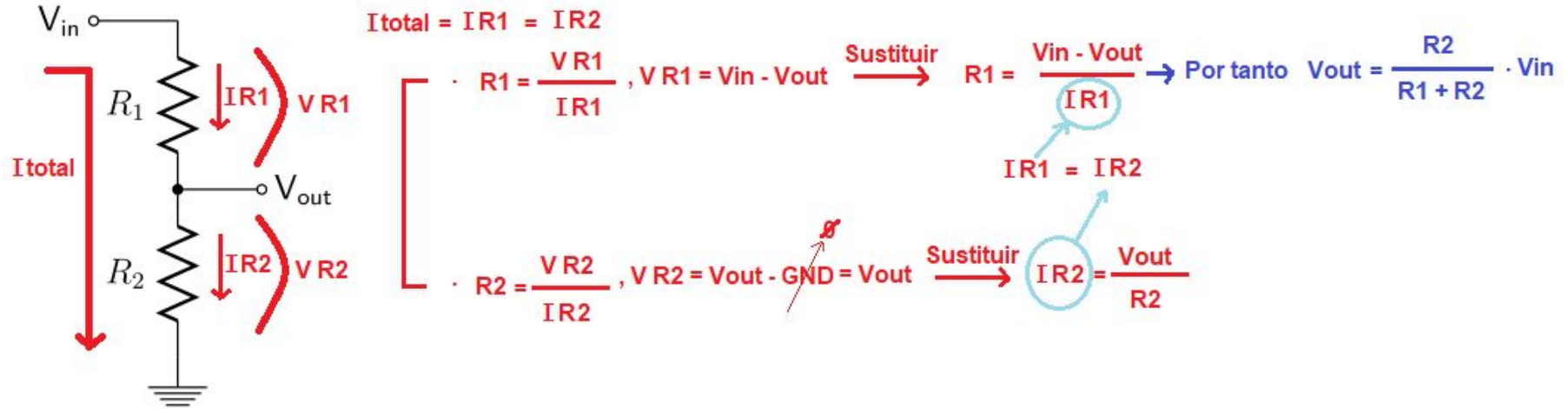
Para un  $V_{in} = 5V$ :

- Si  $R_1 = R_2 = 1 \text{ k}\Omega$ ,  $V_{out}$  será la mitad, **2.5 V**.
- Si  $R_1 = 1 \text{ M}\Omega$ ,  $R_2 = 1 \text{ k}\Omega$ ,  $V_{out}$  será  $\approx 0 \text{ V}$ .
- Si  $R_1 = 0 \text{ }\Omega$ ,  $R_2 = 1 \text{ k}\Omega$ ,  $V_{out}$  será **5 V**.



# Fórmula divisor de voltaje

La fórmula del divisor se halla de la siguiente forma:



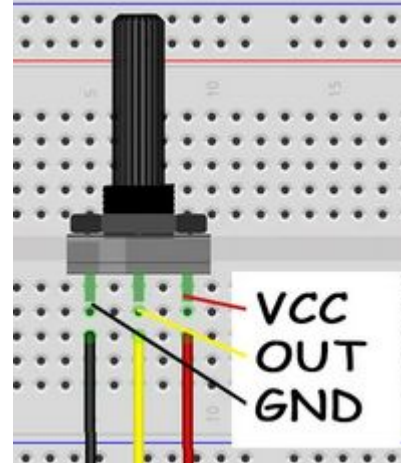
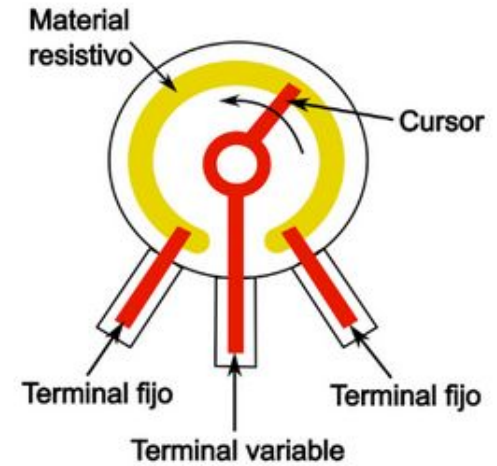
# Actividad

Leer valores potenciómetro:

- 1º pata a 5V
- 3º pata a GND
- 2º pata a entrada analógica

Funciones:

- Serial.begin
- pinMode
- analogRead
- Serial.println





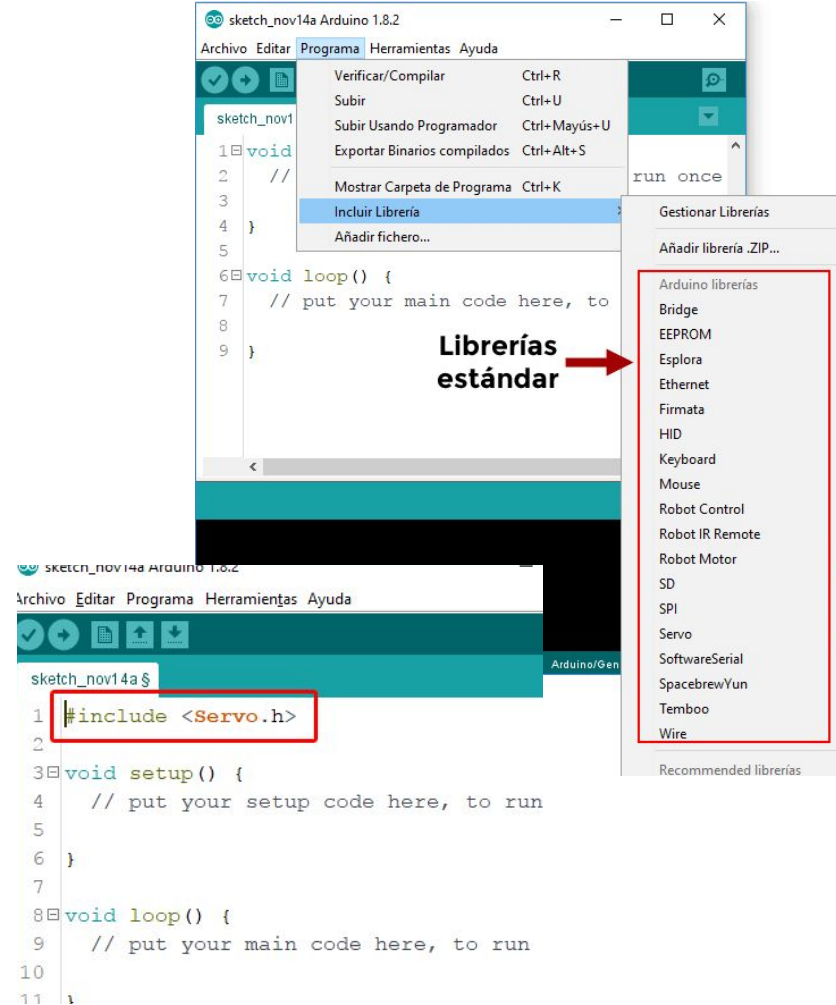
# Librerías

Conjuntos de código ya escrito. Organizado en funciones o variables.

- Estándar: Librerías ya incluidas en Arduino
- Terceros: Librerías creada por gente

Para su uso, se ha de incluir con  
`#include <nombre_de_librería>`

Facilita la reutilización de código ya creado



# Servo

Su control se basa en modulación de ancho de pulso, pero:

Con la ayuda de la librería Servo, se nos facilita el control de los mismos:

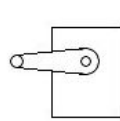
```
#include <Servo.h>
```

```
Servo miServo;
```

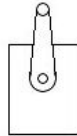
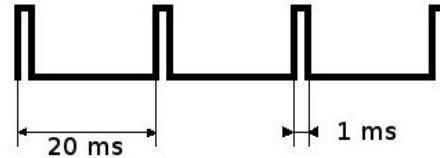
```
miServo.attach(pin);
```

```
miServo.write(90);
```

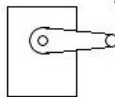
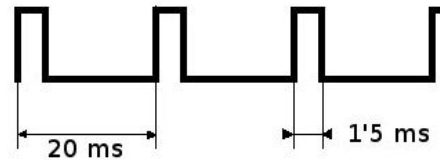
...



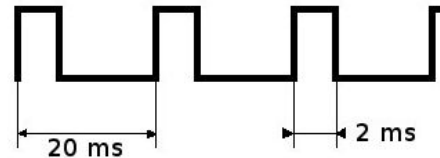
0°



90°



180°



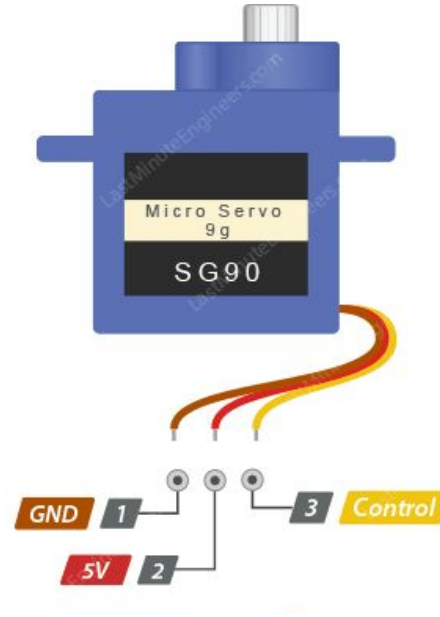
# Actividad

## Rotar servo:

- Cable naranja a salida digital
- Cable marrón a GND
- Cable rojo a 5V

## Funciones:

- Servo servo
- servo.attach
- servo.write



# Actividad

Rotar servo en función de potenciómetro:

- Servo y potenciómetro cableados como prácticas anteriores

Funciones:

- pinMode
- analogRead
- matemáticas
- o map()
- Servo servo
- servo.attach
- servo.write

